基于布尔函数的网络可达性验证方法

张立群，林海涛，沈 钊†

(海军工程大学 电子工程学院，武汉 430000)

摘 要：针对 SDN 网络中由于不同应用的转发路径交叠等导致的数据平面配置问题，提出一种基于布尔函数的网络可达性验证方法。首先，将网络拓扑抽象为端口拓扑并计算端口邻接矩阵；之后，生成网络的路径空间和各端口的转发函数并计算每条路径的路径函数；最后通过判断路径函数的可满足性来确定路径的可达性。通过仿真实验，对网络拓扑和流规则规模等因素对算法验证效率的影响进行研究，并将所提方法与 APV 和 DASDA 进行性能比较。实验结果表明，所提方法能够有效检测 SDN 网络中的流规则配置问题。随着网络中环路的增加和流规则规模的增长，验证网络所需的时间开销逐渐增加。其中，网络拓扑对路径生成时间影响较大，而转发函数的生成时间则主要受流规则规模的影响。所提方法的验证时间相较于 APV 和 DASDA 分别平均缩短约 53.76%和 27.74%。

关键词：软件定义网络；规则配置；可达性验证；布尔函数

中图分类号：TP393.08 doi: 10.19734/j.issn.1001-3695.2022.01.0049

Network reachability verification method based on Boolean functions

Zhang Liqun, Lin Haitao, Shen Zhao†

(*College of Electronics Engineering*, *Naval University of Engineering*, *Wuhan* 430000, *China*)

Abstract: Aiming at the data plane configuration problem caused by overlapping forwarding paths of different applications in SDN network, this paper proposes a Boolean function-based network reachability verification method. First, it abstracts the network topology into port topology and calculates the port adjacency matrix; then generates the path space of the network and the forwarding function of each port, and calculates the path function of each path; finally, it judges the accessibility of path from the satisfiability of path function. Through simulation experiments, this paper studies the influence of factors such as network topology and flow rule scale on the algorithm verification efficiency, and the performance of the proposed method is compared with APV and DASDA. Experimental results show that the proposed method can effectively detect flow rule configuration problems in SDN networks. As the number of loops in the network increases and the size of the flow rules grows, the time overhead required to verify the network gradually increases. Among them, the network topology has a great influence on the path generation time, and the generation time of the forwarding function is mainly affected by the scale of the flow rule. Compared with APV and DASDA, the time to verify is shortened by an average of about 53.76% and 27.74%, respectively.

Key words: software defined network; rule configuration; reachability verification; Boolean functions

0 引言

为了适应新的网络需求，传统网络体系架构经过几十年的发展增加了大量的网络协议，这使得整个网络变得臃肿不堪[1]。为重新定义网络架构，斯坦福大学在 2006 年的 Clean Slate 项目中提出了软件定义网络[2](Software Defined Network，SDN)。SDN 的基本理念是构建一个网络操作系统，使得仅在控制器软件上进行网络编程就能管控所有网络流量。相较于传统网络架构，SDN 架构具有较好的扩展性、强大的灵活性和快速响应性等特点，被广泛应用于物联网和 5G 开发等大规模网络应用[3, 4]。

但 SDN 也还存在着一些问题，可能导致严重的网络安全事故[5]。首先，部署在控制器中的多个应用对同一转发平面进行操作，使得不同应用的转发路径相互交叠，可能出现多应用的不当依赖。这种不当依赖使得真实的转发平面与管理员预期不符，出现转发环路、非法可达路径以及数据阻塞等网络问题，影响网络的安全和转发性能[6]，如图 1(a)所示。其次，在主流的南向协议 OpenFlow 中定义了基于传输层安全性协议(Transport Layer Security，TLS)的 OpenFlow 通道，用于实现控制器对交换机的管理。但由于 TLS 协议的复杂性，管理员一般使用明文对交换机进行配置管理[7]。这使得在 SDN 中发生中间人攻击的概率要大于传统网络[8]。攻击者会利用 OpenFlow 协议实现欺骗拦截，阻止正常的流量传输，或者在交换机中添加新的流规则，控制数据包转发，使网络平面脱离管理员控制[9]，如图 1(b)所示。目前，SDN 中缺乏有效的网络验证机制，难以应对这些控制平面的配置错误和网络攻击[10]。为此，对网络可达性验证的研究具有重要的应用价值。

1 相关工作

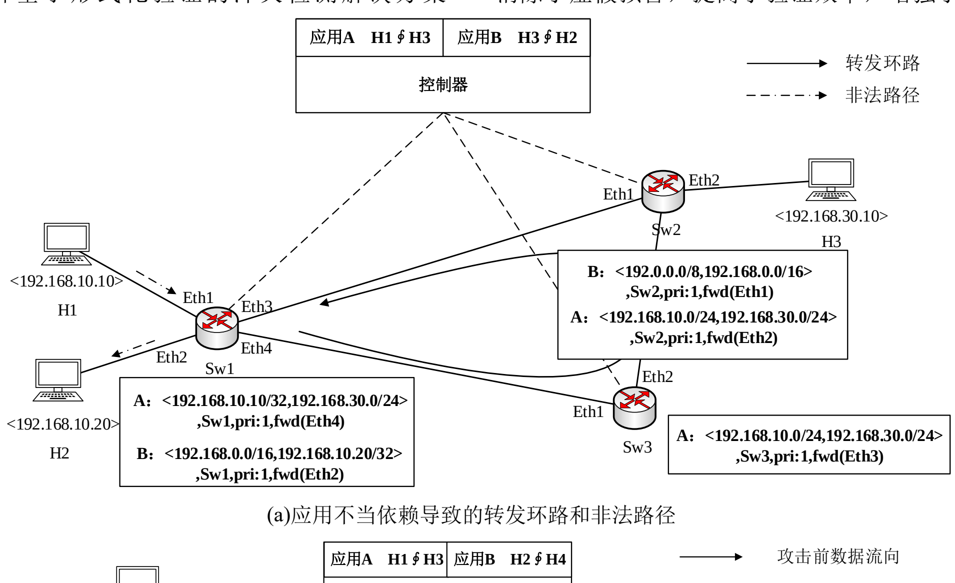
文献[11]提出一种 NetPlumber 工具，NetPlumber 基于头空间分析[12] (Header Space Analysis，HSA)创建并维护流规则之间的依赖关系图，该图将流规则使用管道连接，两流规则的匹配域交集作为管道的过滤器，以增量的方式动态检查状态的合法性变化。文献[13]提出的 NetV 同样是基于 HAS 的网络分析工具。与 NetPlumber 相比，NetV 使用二元决策图解决了通配符表达式在多次并集和差分操作后的爆炸问题，减少了使用不同通配符筛选器执行相同行为规则的冗余。

收稿日期：2022-01-16；**修回日期：**2022-03-08

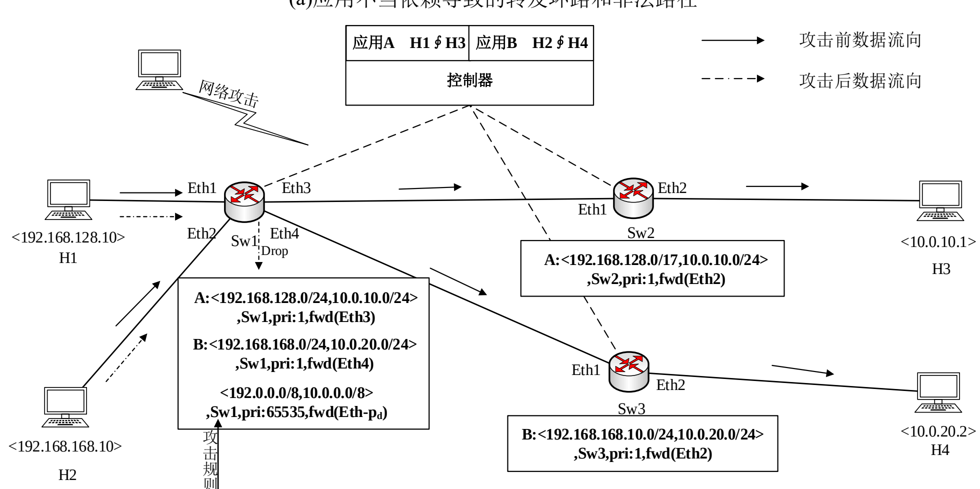
作者简介：张立群(1996-)，男，山东青州人，硕士研究生，主要研究方向为软件定义网络、网络安全；林海涛(1974-)，男，山东潍坊人，副教授，博士，主要研究方向为信息网络管理与规划；沈钊(1980)，男(通信作者)，讲师，硕士，主要研究方向为通信与信息系统(shenzhao_0@163.com)。

文献[14]提出一种高效的流规则冲突检测方法。通过对全网规则的匹配域进行编码，实现对流规则的压缩，进而对全网规则冲突的检测。但由于对规则进行压缩，使得可能在检测过程中出现虚报。文献[15]提出了一种基于原子谓词验证(Atomic Predicates Verifier，APV)的方法，该方法通过将访问控制列表(Access Control Lists，ACL)和转发表(Forward Information dataBase，FIB)构造成端口谓词，并计算谓词集合的原子谓词集。然后使用整数集合来表示各个转发谓词，构建转发图验证网络可达性。但当其网络数目较多时，需要计算大量的原子谓词。并且当原子谓词集合进行更新时，表示端口谓词的整数集合也需要进行调整，验证效率较低。文献[16]提出了一种轻量级的包转发验证方案(Lightweight Packet Forwarding Verification，LPV)，通过对交换机出口和入口的随机采样，比对采样包中的相关信息判断网络中的异常转发。但由于需要不断采样和上传，增加了转发延迟和通信开销。文献[17]提出一种基于形式化验证的冲突检测解决方案DASDA。DASDA 通过底层配置和转发平面快照验证 SDN 拓扑中所有网络设备的行为。交换机中的流规则被抽象为流规则决策图(Flow entries Decision Diagram，FeDD)，FeDD 中的路径节点代表数据包的各个匹配字段，路径终点记录着数据包在网络中的所有动作。当交换机数目较少时，算法检测速度相对较快。文献[18]提出一种方法来验证请求的路径集合会不会发生配置错误。但该方法只对路径集合进行了分析，并没有结合具体的流规则。

针对上述研究存在的虚假预警、验证效率低以及通信开销增加等问题，本文提出一种基于布尔函数的网络可达性验证 (Reachability Verification Based on Boolean Functions, RVBBF)方法，RVBBF 使用控制器中已有的网络拓扑和流规则信息，无须与转发平面进行频繁通信，减小了通信开销；并且该方法使用布尔函数对未压缩的流规则匹配域进行描述，结合物理路径对数据平面中的可达性和转发环路进行检测，消除了虚假预警，提高了验证效率，增强了 SDN 的安全性。



(a)应用不当依赖导致的转发环路和非法路径



(b)网络攻击导致的数据阻塞

图 1　网络场景

Fig. 1　Network scenes

2　网络模型

对于每一个数据包，其都有着确定的源 IP 地址，记作 IP_S 和目的 IP 地址，记作 IP_D。在 IPv4 网络中，网络地址由 32 位比特序列表示，本文将数据包抽象为一个 64 位的比特序列(该方法同样可拓展应用于 IPv6)，其中前 32 位表示 IP_S，后 32 位表示 IP_D。整个数据包空间 Ω 可以表示为

$$\Omega=\{(a_0,\cdots,a_{63})\,|\,a_i\in\{0,1\}\}\tag{1}$$

当数据包到达交换机时，会将数据包的包头信息与流规则进行匹配，并根据规则定义的动作处理数据包。流规则一般包含匹配域、动作域、优先级以及计数器等字段。为简化问题，使得仅通过布尔运算就能判断数据包与流规则的匹配关系以及求解流规则所能匹配的数据包集合，作出如下定义，

定义 1　流规则格式定义如式(2)所示。

$$R:<sw,pri,M,action>\tag{2}$$

其中，sw 记录规则所在的交换机；pri 表示规则的优先级，其取值为[0,65535]内的整数，数值越大则优先级越高；匹配域 $M=(Net_S,Net_D)$，Net_S 和 Net_D 分别表示规则匹配的源地址网络和目的地址网络，两者皆为前缀匹配字段；$action$ 为动作域，表示交换机对数据包的操作。当该数据包到达 sw 时，将按照

所匹配规则的 *action* 进行操作。

定义 2　交换机对数据包的操作包含转发和丢弃两种，对动作域 *action* 使用式(3)描述。

$$action = fwd(Eth) \tag{3}$$

对于转发动作，使用本地端口 *Eth* 描述，表示将数据包从对应端口转发；对于丢弃动作，可认为交换机将数据包从特殊的端口 Eth_d 转发，即 $action = fwd(Eth_d)$ 。

定义 3　对于任意流规则 R ，其可匹配的数据包空间为 Ω_R ，则一定存在一个 64 元的布尔函数 B ，使得

$$\begin{cases} B(\omega) = True, \forall \omega \in \Omega_R \\ \xi \in \Omega_R, \forall \xi \in \{\xi \mid B(\xi) = True\} \end{cases} \tag{4}$$

称 B 为流规则 R 的匹配函数，形式如下：

$$B = v_0 \wedge v_1 \wedge \cdots \wedge v_{63} \tag{5}$$

对于匹配函数存在下列性质：

性质 1　对于数据包 ω_0 和匹配函数 B ，若 $B(\omega_0) = True$ ，则 ω_0 匹配流规则；否则数据包与该流规则不匹配。

性质 2　若 B 为某一流规则的匹配函数，则 $B(\omega) = True$ 的解空间，即为该规则所有可匹配的数据包。

性质 3　存在两个特殊的匹配函数 B_T 和 B_F ，分别有

$$B_T(\omega) = True, \forall \omega \in \Omega \tag{6}$$

$$B_F(\omega) = False, \forall \omega \in \Omega \tag{7}$$

值得注意的是，并非所有匹配函数都具有全部的 64 个比特变量。例如，存在流规则 R ，其匹配域 $M = (192.168.0.0/16\ ,10.0.0.0/8)$ ，则 B_R 如下所示，

$$\begin{aligned} B_R = {} & v_0 \wedge v_1 \wedge \neg v_2 \wedge \neg v_3 \wedge \neg v_4 \wedge \neg v_5 \wedge \neg v_6 \wedge \neg v_7 \wedge \\ & v_8 \wedge \neg v_9 \wedge v_{10} \wedge \neg v_{11} \wedge v_{12} \wedge \neg v_{13} \wedge \neg v_{14} \wedge \neg v_{15} \wedge \\ & \neg v_{32} \wedge \neg v_{33} \wedge \neg v_{34} \wedge \neg v_{35} \wedge v_{36} \wedge \neg v_{37} \wedge v_{38} \wedge \neg v_{39} \end{aligned}$$

在 B_R 中只包含 24 个比特变量，这是由于 Net_S 和 Net_D 的网络前缀分别为 16 和 8。对于 Net_S 来说，需要从 v_0 到 v_{15} 的 16 个比特变量即可表示；同理对于 Net_D 则只需从 v_{32} 到 v_{39} 的 8 个比特变量。

本文使用二元决策图[19](Binary Decision Diagram, BDD)来对布尔函数进行表示。BDD 是用于表示布尔函数的有向无环图，可以极大的减小验证布尔函数满足性需要的时间开销，其结构如图 2 所示。

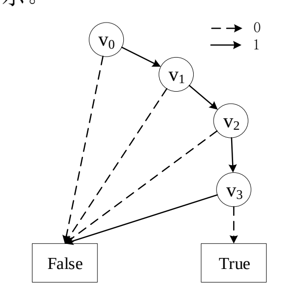


图 2　函数 $B = v_0 v_1 v_2 \neg v_3$ 的 BDD

Fig. 2　The BDD of　$B = v_0 v_1 v_2 \neg v_3$

3　基于布尔函数的网络可达性验证

为解决 SDN 网络中缺乏可达性验证机制的问题，基于上一节的网络模型提出一种基于布尔函数的网络可达性验证方法。RVBBF 的工作流程包括四部分：抽象端口拓扑、计算转发函数、生成路径空间和路径验证，其流程结构如图 3 所示。

RVBBF 对流规则和数据包的匹配关系用布尔函数进行描述，并使用路径搜索算法对转发平面的拓扑结构进行搜索生成路径空间，将网络可达性问题转换为布尔可满足性问题(Boolean Satisfiability Problem，SAT)，使得仅通过简单的逻辑运算就能够验证网络端到端的可达性。为提高验证效率，RVBBF 对生成路径空间和计算转发函数两阶段做并行处理，使二者同时进行；并且使用 BDD 结构表示方法中的布尔函数，有效缩短路径验证时间。

下面将对 RVBBF 的各阶段工作进行详细描述。

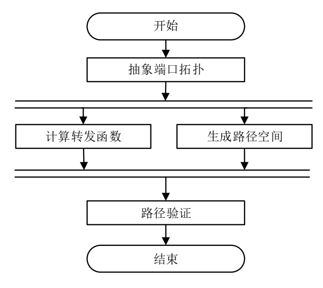
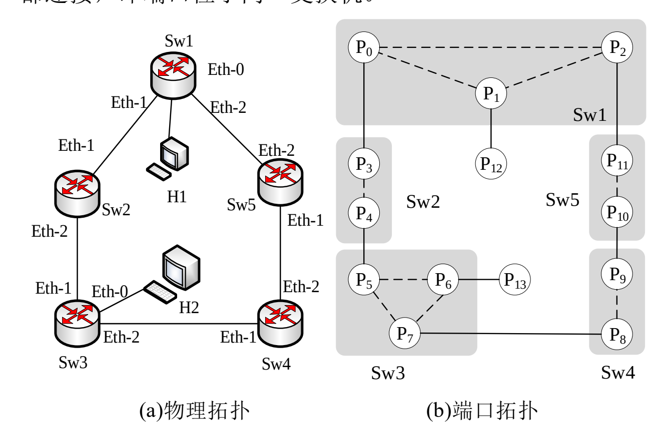


图 3　工作流程结构

Fig. 3　The method flow structure

3.1　抽象端口拓扑

对网络的物理拓扑进行抽象，可得网络的端口拓扑，如图 4 所示。在端口拓扑中，节点表示设备端口，分为交换机节点和主机节点；边则表示端口之间的连接关系，实线为外部连接，即端口位于不同交换机且存在物理链路；虚线为内部连接，即端口位于同一交换机。



(a)物理拓扑　　(b)端口拓扑

图 4　端口拓扑抽象

Fig. 4　Port abstraction

为方便后续研究，将 $<sw,eth>$ 映射为全局端口 P ，映射规则记为 D 。例如在上述拓扑中有，

$$D[<\mathrm{Sw1,Eth\text{-}0}>] \to \mathrm{P_1}$$

$$D[<\mathrm{Sw1,Eth\text{-}1}>] \to \mathrm{P_0}$$

$$\vdots$$

由 D 和各端口的连接关系，可得网络拓扑的邻接矩阵 $\boldsymbol{E}$ ，其中 $e_{ij} \in \{-1,0,1\}$ 表示端口 P_i 和 P_j 的连接关系。若 $e_{ij} = -1$，表示两节点为内部连接；若 $e_{ij} = 0$ ，表示两节点无连接；若 $e_{ij} = 1$，则表示两节点为外部连接。上述拓扑的邻接矩阵如下所示：

$$\boldsymbol{E} = \begin{bmatrix} 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & & & & & \ddots & & & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.2　计算转发函数

匹配函数 B 可以求得流规则所匹配的数据包，但在交换机内部存在大量的流规则，并且优先级和转发端口各异，难以用其表征交换机的转发动作。为此类比匹配函数，对转发函数进行如下定义：

定义 4　对于任意的交换机端口 P ，通过其转发的数据包空间为 Ω_P ，则一定存在一个 64 元的布尔函数 F_P ，使得

$$\begin{cases} F_P(\omega) = True, \forall \omega \in \Omega_P \\ \xi \in \Omega_P, \forall \xi \in \{\xi \mid F_P(\xi) = True\} \end{cases} \tag{8}$$

称布尔函数 F_P 为端口 P 的转发函数。

根据流规则集合和 D 可计算任意端口的转发函数 F_P ，其

步骤如下：

首先，通过算法 1，对获取的流规则进行预处理，计算其匹配函数，并将端口映射为全局端口，得到使用匹配函数表示的流规则 $< sw, pri, B, P >$。

算法 1　流规则预处理

输入：$< sw, pri, M, action >$，D。

输出：$< sw, pri, B, P >$。

$M' \leftarrow \mathrm{tran}(M)$
$B \leftarrow B_T$;
for $i \leftarrow 1$ **to** $\mathrm{len}(M')$ **do**
　switch($M'[i]$)
　　case 0:
　　　$B \leftarrow B \wedge v_i$;
　　case 1:
　　　$B \leftarrow B \wedge \neg v_i$;
　　case *:
　　　Skip;
　end switch
end for
$Eth \leftarrow$ get_Eth(action)
$P \leftarrow D[< sw, Eth >]$
return $< sw, pri, B, P >$

在算法 1 中，函数 tran() 将 M 转换为三元序列 M'，

$$M' \in \{(m_0, \cdots, m_{63}) \mid m_i \in \{0,1,*\}\}$$

其中，$(m_0, \cdots, m_{31})$ 表示 Net_S，$(m_{32}, \cdots m_{63})$ 表示 Net_D，* 为通配符。若 M' 的第 i 位为 1 或 0，取位变量 v_i 或 $\neg v_i$；若第 i 位为 *，则跳过位变量 v_i。所有位变量的析取即为该流规则的匹配函数。get_Eth() 则根据 *action* 获取本地转发端口 Eth，之后根据 D 将 $< sw, Eth >$ 映射为 P。

数据包在交换机内进行规则匹配时，是按照优先级进行的，数值越大，优先级越高。为实现对同一交换机中高优先级的流规则优先处理，将规则以 (sw, pri) 为关键字进行排序，其中 sw 为主关键字，进行升序排列，pri 为次关键字，进行降序排列。可得到有序的流规则列表，如下所示，

$$\begin{gathered} < sw_1, pri_1, B_1, P_1 > \\ \vdots \\ < sw_j, pri_j, B_j, P_j > \\ \vdots \\ < sw_m, pri_m, B_m, P_m > \end{gathered}$$

其中，对于相同 sw 的规则,若 $j < k$，有 $pri_j \geq pri_k$。

最后通过算法 2，根据有序的规则列表和端口集合，计算各个端口的转发函数。

算法 2　计算转发函数

输入：有序的流规则列表，端口集合 $\{1, 2, \cdots, n\}$。

输出：转发函数集合 $\{F_1, \cdots, F_n\}$。

for $j \leftarrow 1$ **to** n **do**
　$F_j \leftarrow B_F$;
end for
$fd \leftarrow B_F$;
$s \leftarrow sw_1$;
$i \leftarrow 1$;
while $i \leq m$ **do**
　if $sw_i = s$ **then**
　　$F_{P_i} \leftarrow F_{P_i} \vee (B_i \wedge \neg fd)$
　　$fd \leftarrow fd \vee B_i$
　　$i \leftarrow i + 1$
　else
　　$fd \leftarrow B_F$
　　$s \leftarrow sw_i$
　　$i \leftarrow i$
　end if
end while
return $\{F_1, F_2, \cdots, F_n\}$

3.3　生成路径空间

按照真实网络中的转发模式，数据包一般是从交换机的某一端口 P_M 进入，通过内部连接到转发端口 P_N，然后从 P_N 通过外部连接转发至另一交换机的端口 P_X 进入交换机，不断重复，如图 5 所示。为此，根据邻接矩阵 $\boldsymbol{E}$ 和起始端口 P，可生成 P 的路径空间 Set_P。

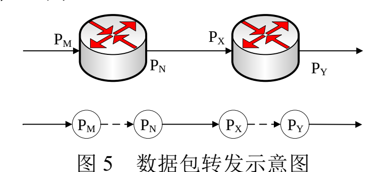


图 5　数据包转发示意图

Fig. 5　Schematic diagram of packet forwarding

路径空间中的基本元素是物理可达路径 *path*，并具有以下特点：a) *path* 以某一主机节点作为起始节点，通过外部连接与交换机节点相连，而后通过内部连接与同一交换机的节点相连，交替进行；b)当访问到已访问过交换机的节点时，*path* 结束；c)当访问到已访问过的节点时，*path* 结束；d)当访问到主机节点时，*path* 结束。

通过基于深度优先搜索(Depth First Search，DFS)的算法 3，可由拓扑的邻接矩阵和起始节点，生成路径空间 Set_P。

算法 3　生成路径空间

输入：邻接矩阵 E，起始节点 P。

输出：路径空间 $Set_P = \{path_1, \cdots path_k\}$。

$now = P$; $Set_P \leftarrow \varnothing$;
$path \leftarrow [now]$;
while *True* **do**
　while *True* **do**
　　if find_next(*now*) **then**
　　　$now \leftarrow$ find_next(*now*);
　　else
　　　break;
　　end if
　　path.append(*now*)
　　if *now* **or** (*now.Sw* **is** visited) **then**
　　　break;
　　end if
　end while
　$Set_P \leftarrow Set_P \cup \{path\}$
　while continue_back() **do**
　　path.pop()
　　if $len(path) = 0$ **then**
　　　break;
　　end if
　　$now \leftarrow path[-1]$
　end while
　if find_over() **then**
　　break
　end if
end while
return Set_P

其中，函数 find_next() 可以返回当前正在访问节点 *now* 可达但

尚未访问的下一跳节点；当 now 的所有可达节点都被访问过后，continue_back() 返回 $True$，路径回溯；若 $now=P$ 且 now 的所有可达节点都已访问，则所有路径都已经找到，函数 check_over() 返回为 $True$，算法结束并返回路径空间 Set_P。

3.4　路径验证

Set_P 中的 $path$ 只是物理可达并不一定逻辑可达，即交换机不一定存在相应的流规则来转发数据包。通过算法 4，对所有的物理可达路径进行验证，在这个过程中，可以检测路径包含的环路以及可达性。

算法 4　路径可达性验证

输入：路径空间 Set_P，转发函数集 $\{F_1,\cdots,F_n\}$。

输出：循环路径空间 Cir、可达路径空间 Rea。

```
for path in Set_P do
    B_pt ← B_T ;
    for P in path do
        if P is Output then
            B_pt ← B_pt ∧ F_P ;
        end if
    end for
    if SAT ( B_pt ) then
        if path.tail in path − path.tail then
            Cir ← Cir ∪ {path} ;
        else
            Rea ← Rea ∪ {path} ;
        end if
    end if
end for
return Cir , Rea
```

算法 4 通过遍历路径空间，对路径中输出端口的转发函数 F 进行合取得到路径函数 B_{pt}，然后对 B_{pt} 进行 SAT 验证。若存在可满足的解，则存在数据包能够同时通过路径上所有的转发端口，其解空间就是可通过该路径的数据包集合，路径存在逻辑通路。之后再对路径属性进行判断，若路径的终端节点在路径中重复出现，则表明该路径是循环路径，否则为单向可达路径。若解不存在，说明不存在数据包可以通过该条路径上所有转发端口，即路径逻辑不可达。

4　实验分析

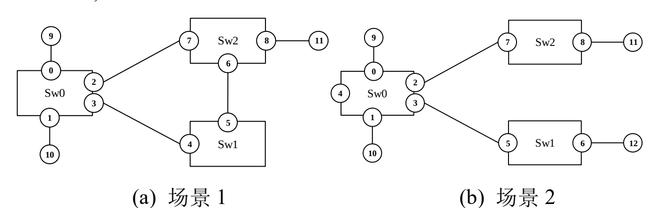
4.1　实验条件

本文实验计算机的配置如下：Intel Core i7-9750H CPU 2.6GHz 处理器，16GB 内存，Win10 操作系统。在 Visual Studio Code 平台上基于 Python 语言对该方法进行实现。在性能分析实验中不同规模的流规则集合为通过使用 Classbench 工具随机生成。

4.2　实验结果与分析

4.2.1 有效性分析

为验证 RVBBF 的有效性，对引言中所提的网络场景进行分析验证，其端口拓扑(省略内部连接)及映射关系如图 6 所示。以 H1(在两图中均为节点 9)为起始节点生成路径空间，两场景的路径空间如表 1 所示(加粗路径表示逻辑可达)。



(a) 场景 1　　(b) 场景 2

图 6　示例场景的端口示意

Fig. 6　Port representation of the example scene

表 1　不同场景的路径空间

Tab. 1　Path space for different scenes

Scene Number	Path space
Scene 1	**[9,0,1,10]**
	[9,0,3,4,5,6,7,2]
	[9,0,3,4,5,6,8,11]
	[9,0,2,7,6,5,4,3]
	[9,0,2,7,8,11]
Scene 2	**[9,0,4]**
	[9,0,1,10]
	[9,0,2,7,8,11]
	[9,0,3,5,6,12]

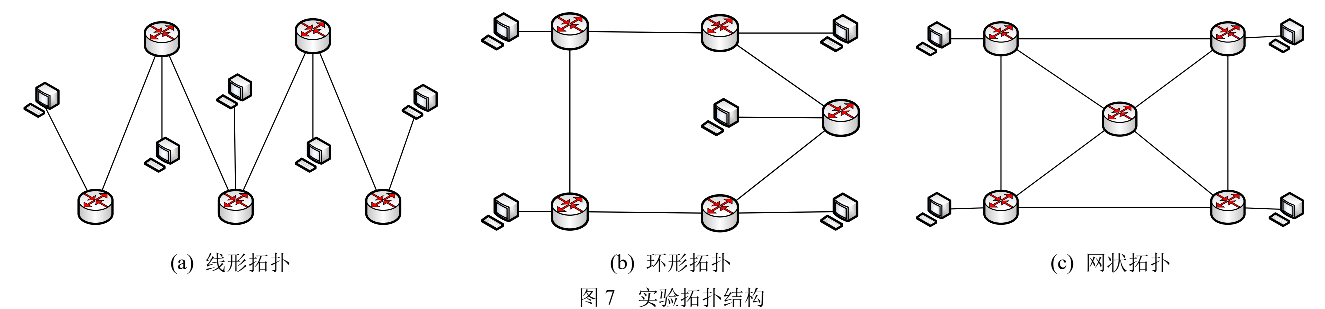
结合转发函数对表 1 路径进行验证，发现场景 1 中存在一条转发环路[9,0,3,4,5,6,7,2]和一条非法可达路径[9,0,1,10]；在场景 2 中则只有一条数据通路[9,0,4]，其中节点 4 为 $<Sw0,Eth_d>$，表明 Sw1 可能受到攻击或规则设置错误，造成数据阻塞。结果表明，RVBBF 能够有效检测网络中由于应用的不当依赖和网络攻击导致的流规则冲突。

4.2.2 性能分析

为验证 RVBBF 的性能，本小节通过改变网络拓扑和流规则数目等参数，对算法各部分验证效率进行分析。为提高准确性，本节数据均为多次实验取平均值。

1) 网络拓扑对生成路径空间的影响

在实验中设置 5 台交换机，分别组成线形、环形和网状 3 种典型的基本网络拓扑，如图 7 所示。通过改变不同拓扑下的终端主机数量，研究网络拓扑对路径空间生成时间的影响，实验结果如图 8。各个拓扑的路径空间(由于终端主机地位对等，任取其一作为起始节点)大小如表 2 所示。



(a) 线形拓扑　　(b) 环形拓扑　　(c) 网状拓扑

图 7　实验拓扑结构

Fig. 7　Experimental topology

由表 2 和图 8 可以看出，在相同拓扑下，随着主机数量的增多，路径数量和生成时间也逐渐增加，且不同拓扑之间的差距逐渐增大。网状拓扑的路径空间生成时间要明显高于线形拓扑和环形拓扑。这是由于在网状拓扑中具有较多可达环路，增加相同数目的主机使得增加的物理可达路径会更多，在生成路径空间时需要消耗较多的时间。在相同交换机数目

的情况下，环形拓扑和网状拓扑的路径生成时间平均约为线形拓扑的 3.02 倍和 10.83 倍。

表 2　不同拓扑中的路径数量

Tab. 2　Number of paths in different topologies

Number of hosts	Linear	Ring	Mesh
10	9	19	102
20	19	37	171
30	29	55	232
40	39	73	301
50	49	91	362
60	59	109	431
70	69	127	492
80	79	145	561
90	89	163	622
100	99	181	691

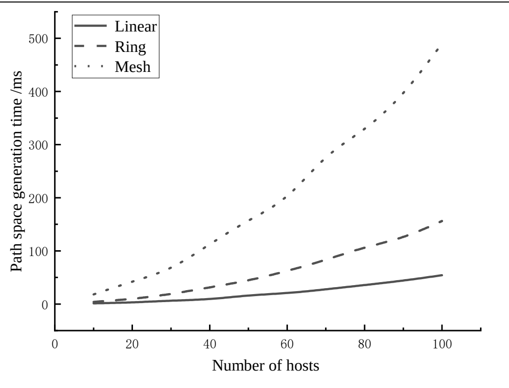


图 8　主机数量对路径空间生成时间的影响

Fig. 8　Influence of the number of hosts on the path generation time

2) 流规则规模对转发函数生成的影响

随机生成不同数量的流规则集合，研究流规则规模对转发函数生成的影响，结果如图 9 所示。

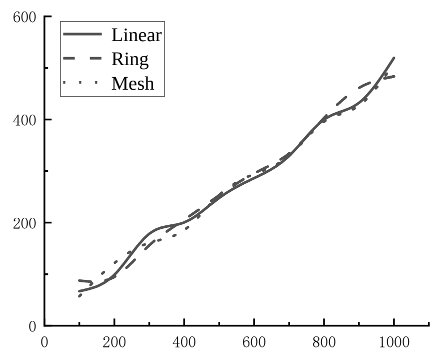


图 9　流规则规模对转发函数生成时间的影响

Fig. 9　Influence of flow rule size on forwarding function generation time

由图 9 可以看出，三种拓扑的转发函数生成时间在相同规则规模下几乎相等，且随着规模的增加而增加。这是由于该阶段需要对预处理后流规则的匹配函数进行析取、合取和求反等操作。每一条规则进行的操作相似。为此，端口函数的生成时间会随着流规则的规模逐渐增加，且基本成线形关系。

3) 流规则数目对路径验证的影响

为避免路径空间大小对路径验证时间的影响，实验设置具有 68 台终端的线形拓扑、37 台终端的环形拓扑和 4 台终端的网状拓扑，其路径空间中都包含 67 条路径。实验结果如图 10 所示。

由图 10 可看出，总体上来说网状拓扑的路径验证时间最长，其次是环形拓扑，线形拓扑的路径验证时间最短。这是因为网状拓扑中的路径相对较长，生成路径函数时需要对更多的转发函数进行析取操作，为此验证时间更长。而且由于使用了 BDD 对布尔函数进行存储，极大缩短了对路径的验证时间。在上述实验中，路径验证时间均不超过 20ms。

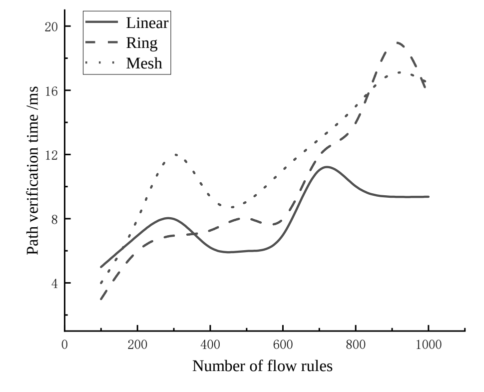


图 10　流规则规模对路径验证时间的影响

Fig. 10　Influence of flow rule size on path verification time

4.2.3 与 APV 算法的性能比较

本小节将对 RVBBF 与文献[15]提出的 APV 和文献[17]提出的 DASDA 进行性能比较。实验在如图 11 所示的复杂拓扑下进行，通过改变流规则的集合规模，分析其验证效率，实验结果如图 12 所示。

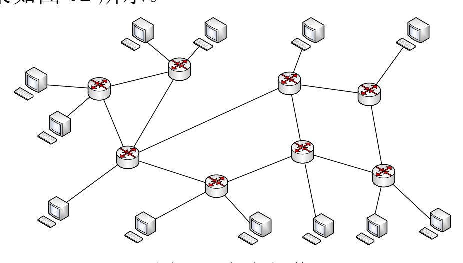


图 11　复杂拓扑

Fig. 11　Complex topology

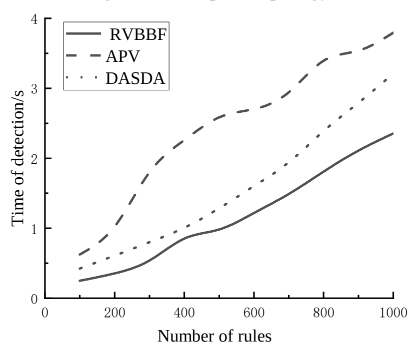


图 12　与 APV 和 DASDA 的性能比较

Fig. 12　Performance comparison with APV and DASDA

由图 11 可以发现，三种算法的检测时间都随着流规则规模的增加而增加，但 RVBBF 在验证效率上要优于 APV 和 DASDA，验证时间分别平均缩短约 53.76%和 27.74%。这是由于随着流规则数目的增加，尤其是当流规则集合包含的网段较多时，APV 中的原子谓词数量会显著增加，DASDA 则会因此导致 FeDD 过于庞大，使得在对其进行维护和查询时需要更多的时间。并且在 APV 中，新谓词的出现可能会对当前的原子谓词集合造成影响，导致其需要重新计算原子谓词集合，增加了验证的时间消耗。相对于 APV，DASDA 没有复杂的原子谓词计算过程，检测效率相对较高。而 RVBBF 则只需计算交换机各端口的转发函数，再据此对路径空间中的路径进行验证即可。同时由于使用 BDD 对布尔函数进行表示，大大提高了 RVBBF 的路径验证验证效率。

5　结束语

针对 SDN 网络中由于不同应用的转发路径交叠等导致的数据平面配置问题，本文提出了一种基于布尔函数的网络可达性验证方法。该方法根据流规则生成转发函数，结合路

径生成算法产生的路径空间可实现 SDN 数据转发平面的可达性验证。实验结果表明，RVBBF 能够有效验证网络可达性。并且相比较 APV，其验证效率有明显提高。

在当今网络中，转发平面中的配置错误非常普遍。新技术也带来了新的安全挑战。在经典的 SDN 南向接口协议 OpenFlow 中，定义的动作中存在能够修改数据包 IP 地址的 SET 动作，该动作可以对数据包的包头信息进行修改，使得网络管理员方便、灵活地实现数据牵引、路由重定向等功能。但如果对 SET 动作使用不当，则会造成防火墙失效，引发严重的网络安全事故[20]。因此下一步将就如何快速高效的检测由 SET 动作导致的转发平面威胁展开研究，进一步提高网络安全。

参考文献：

[1] 杨泽卫，李呈. 重构网络: SDN 架构与实现 [M]. 北京：电子工业出版社，2017: 1-16. (YANG Z W, LI C. Refactoring networks: SDN architecture and implementation [M]. Beijing: Publishing House of Electronics Industry, 2017: 1-16.)

[2] GREENBERG A, HJALMTYSSON G, MALTZ D A, *et al.* A clean slate 4D approach to network control and management [J]. ACM SIGCOMM Computer Communication Review, 2005, 35 (5): 41-54.

[3] MAITY I, MONDAL A, MISRA S, *et al.* Tensor-based rule-space management system in SDN [J]. IEEE Systems Journal, 2018, 13 (4): 3921-8.

[4] Barakabitze A A, Ahmad A, Mijumbi R, *et al.* 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges [J]. Computer Networks, 2020, 167: 106984.

[5] 刘艺，雷程，张红旗，等. 基于 MapReduce 的 OpenFlow 网络属性验证技术 [J]. 计算机研究与发展，2016, 53 (11): 2500-11. (LIU Y, LEI C, ZHANG H Q, *et al.* MapReduce-based network property verification technique for OpenFlow network [J]. Journal of Computer Research and Development, 2016, 53 (11): 2500-11.)

[6] 王军. SDN 下的策略冲突检测研究 [D]. 成都：电子科技大学，2020. (WANG J. Research on Strategy Conflict Detection under SDN [D]. Chengdu: University of Electronic Science and Technology of China, 2020.)

[7] BENTON K, CAMP L J, SMALL C. OpenFlow vulnerability assessment [C]// Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Hong Kong, Aug 16, 2013. New York: ACM, 2013: 151-152.

[8] 董仕. 软件定义网络安全问题研究综述 [J]. 计算机科学，2021, 48 (03): 295-306. (DONG S. Survey on Software Defined Networks Security [J]. Computer Science, 2021, 48 (03): 295-306.)

[9] 郭中孚，张兴明，赵博，等. 软件定义网络数据平面安全综述 [J]. 网络与信息安全学报，2018, 4 (11): 1-12. (GUO Z F, ZHANG X M, ZHAO B, *et al.* Survey of software-defined networking data plane security [J]. Chinese Journal of Network and Information Security, 2018, 4 (11): 1-12.)

[10] 陈浩宇，邹德清，金海. 面向 SDN/NFV 环境的网络功能策略验证 [J]. 网络与信息安全学报，2021, 7 (03): 59-71. (CHEN H Y, ZHOU Q D, JIN H, *et al.* Verification on policies for network functions in SDN/NFV-based environment [J]. Chinese Journal of Network and Information Security, 2021, 7 (03): 59-71.)

[11] KAZEMIAN P, CHANG M, ZENG H, *et al.* Real time network policy checking using header space analysis [C]// Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, Illinois, Apr 2-3, 2013. New York: ACM Press, 2013: 99-111.

[12] KAZEMIAN P, VARGHESE G, MCKEOWN N. Header space analysis: Static checking for networks [C]// Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, Illinois, Apr 25-27, 2012. New York: ACM Press, 2012: 113-126.

[13] Fang Y, Lu Y. Real-Time Verification of Network Properties Based on Header Space [J]. IEEE Access, 2020, 8: 36789-36806.

[14] 郝巍，伊鹏，江逸茗. 一种快速的 SDN 规则冲突检测机制 [J]. 计算机工程，2019, 45 (02): 139-43. (HAO W, YI P, JIANG Y M. A Fast SDN Rule Conflict Detection Mechanism [J]. Computer Engineering, 2019, 45 (02): 139-43.)

[15] YANG H, LAM S S J I A T O N. Real-time verification of network properties using atomic predicates [J]. IEEE/ACM Transactions on Networking, 2015, 24 (2): 887-900.

[16] 王首一，李琦，张云. 轻量级的软件定义网络数据包转发验证 [J]. 计算机学报，2019, 42 (01): 176-89. (WANG S Y, LI Q, ZHANG Y. LPV: Lightweight packet forwarding verification in SDN [J]. Chinese Journal of Computers, 2019, 42 (01): 176-89.)

[17] Saied W, Souayeh N B Y B, Saadaoui A, *et al.* Deep and automated SDN data plane analysis [C]// 2019 International Conference on Software, Telecommunications and Computer Networks, Split, Sept 19. 2019. IEEE, 2019: 1-6.

[18] Burdonov I, Kossachev A, Yevtushenko N, *et al.* Preventive Model-based Verification and Repairing for SDN Requests [J]. arXiv preprint arXiv: 1906. 03101, 2019.

[19] Bryant R E. Graph-based algorithms for boolean function manipulation [J]. Computers, IEEE Transactions on, 1986, 100 (8): 677-691.

[20] Saâdaoui A, Souayeh N B Y B, Bouhoula A. Automated and Optimized Formal Approach to Verify SDN Access-Control Misconfigurations [C]// International Conference on Testbeds and Research Infrastructures, Shanghai, December 1-3, 2018. Cham: Springer Press 2018: 96-112.